# METHOD AND APPARATUS OF RETAINING MAXIMUM SPEED OF FLIP-FLOP METASTABILITY BASED RANDOM NUMBER GENERATORS

**[0001]** This application claims priority to provisional patent application Serial Number 60/454,835 filed March 14, 2003.

## FIELD OF THE INVENTION

**[0002]** The present invention relates to use of metastable states of flip-flops as a basis for random number generation. More particularly, the present invention relates to a method and apparatus for keeping the speed of random number generators utilizing metastability of flip-flops at a maximum.

## BACKGROUND ART

**[0003]** It is known that when the hold and setup times of flip-flops (such as D-type flip-flops) are violated, the flip-flop often enters a metastable state. Metastability can also occur when both inputs to a latch are either both 00 or 11.

**[0004]** Metastability can cause the latch outputs to oscillate unpredictably in a statistically known manner. While theoretically it is possible for the latch outputs to oscillate in a statistically known manner, in reality the latch will randomly shift and arrive at random output values. Such metastable values are then detected by other circuitry as different logic states.

**[0005]** Previous work by the instant inventor has focused on using the unpredictability, or randomness, of the metastable flip-flop, to provide a true random number generator.

**[0006]** Fig. 1 illustrates a random number generator comprising a latch 150 having two cross-connected NAND gates 110 and 115. The flip-flop 105 receives its clock pulse from clock oscillator 112, and the inverted output (-Q) is fed back to the D input, shaping the clock signal to square wave. The non-inverted output (Q) is then fed to delay devices 113, 114, respectively. Each of the delay devices has an output directly in series with one of the NAND gates, 110,115.



Attorney Docket No.: US03 0059
Express Mail No. EV 417136670 US

**[0007]** If the composition of the two NAND Gates 110,115 were exactly the same, there would be no need for the delay devices to achieve the highest frequency to get flip-flop 150 to become metastable. However, the NAND gates will ordinarily differ somewhat, and their differences (gain, offset, speed...) will influence their metastability.

**[0008]** In addition, the difference between the NAND gates changes with temperature, supply voltage and possible environmental factors, so for the highest frequency at which the flip-flop gets metastable, one of the delays has to be tuned dynamically in very small steps. If the flip-flop gets metastable, the output signal is random. However, although the output is random, it is usually not even at a standard logic level.

**[0009]** Fig. 2 shows a string of conventional cascaded flip-flops 205,210,215. We know from this prior art arrangement that even these few cascaded flip-flops ensures with a very high probability that the output is in a 0 or 1 level. If the input to D in flip-flop 205 should come from a metastable flip-flop, the output of this detector is sufficiently random.

**[0010]** However, when one measures the random number generation speed, it is often unknown whether the optimum delay is equal to, smaller or larger than the actual delay. Thus, a complex algorithm is needed to find the highest speed and keep the circuit at this point.

**[0011]** With regard to Fig. 1, it is easy to see that one of the delay elements 113, 114 can be fixed to produce a signal delay equal to the median of the delay range necessary to keep the speed at a maximum. The other delay element is enough to be variable within the delay range so as to provide a large number of different delay values (typically 16-256), and can be selected by suitable codes (e.g. a binary multiplexer). However, in order to maximize such a system, there is a need for a method that dynamically tunes the delay to keep the random number generator at maximum speed.

## SUMMARY OF THE INVENTION

**[0012]** The present invention is directed to a method and apparatus and system that tracks the frequency of the occurrences of metastability and makes adjustments to optimize the value of one of the delay devices. Thus, the random number generator is kept at an

optimum speed by the dynamic tuning of one of the delay devices as environmental factors change. A module measures the speed of the random bit generation and dynamically tunes one of the delay units according to an algorithm.

## BRIEF DESCRIPTION OF THE DRAWING

**[0013]** Fig. 1 is a prior art illustration featuring a randomness source for a random number generator.

**[0014]** Fig. 2 is a prior art illustration of cascaded flip-flops that, with very high probability produce stable output at metastable inputs.

**[0015]** Fig. 3 is an illustration of the present method.

**[0016]** Fig. 4 is a flowchart providing a general overview of one way a method according to the present invention can be performed.

## DETAILED EMBODIMENTS

**[0017]** Fig. 3 shows an illustration of an apparatus according to the instant invention. It should be understood that there are various modifications that a person of ordinary skill in the art could make that do not depart from the spirit of the invention or the scope of the appended claims.

**[0018]** The input 301 would be from a square wave source. The metastability, as previously discussed, can be caused by, for example, violating setup and hold times of a flip-flop or applying forbidden input combination to a dual input latch so as to create the metastable output of the circuit at 330. The input 301 is fed to both a fixed delay unit 305 and a variable delay unit 310. The outputs of the respective delay units are fed to an input of respective NAND gates 315, 320. Whereas the output of NAND gate 320 is fed to an input of NAND gate 315, the output of NAND gate 315 is fed to an input of NAND gate 320 and it is provided to a frequency measurement-, delay tuning - module 312. The delay of the delay element 310 is dynamically tuned affecting the input signal of NAND gate 320. The module 312 monitors the output 330 to determine the frequency how often the flip-flop formed by the NAND gates 315 and 320 gets metastable and dynamically tunes the variable

delay unit 312 to maximize it to optimize the speed of the random number generator utilizing the signal 330.

**[0019]** There are many ways known in the art, how random numbers can be generated from the random occurrences of metastability seen at the output of the circle at 330. These include the final logic value to which the metastable circuit resolves to, the time point at which the metastability occurs, the length of the metastable event, etc.

**[0020]** Although the present inventor recommends as a best mode that the fixed delay be set to perform a signal delay equal to the center of the delay range necessary to keep maximum speed, it is clearly within the spirit and scope of the invention that the fixed delay could be set at other values, which in turn, would require the variable delay unit 310 to be set at different values.

**[0021]** With regard to the frequency measurement delay tuning module 312, a microprocessor is best applied for the speed measurements of the random bit generation and for the delay tune algorithm. However, the functions of module 312 can also be provided for by implementing specially designed hardware.

**[0022]** For example, the frequency measurement can be performed with a counter, which is multiplied by a weight (<1) at every clock cycle and incremented by 1 each time a random bit is produced. The following C code is provided merely for purposes of illustration and not for limitation as one way that the apparatus can operate. It should be understood that other ways or different types of code could be used:

```
#define Weight 0.9990234375 // 1 – 1/(2<<10)
Counter = Counter * Weight + IsRandomBitGenerated();
```

**[0023]** Thus, the current floating point value of the counter is related to the speed of the random bit generation during the last clock cycles. With the above constant it is a few thousand clocks. The multiplication with the weight can be replaced by periodically shifting an integer counter one bit to the right (Divide by 2). It should be understood, that the frequency measurement of the occurrences of metastability can be done by any other dynamic frequency measurement algorithm (having a finite or infinite length integrator) known in the art.

**[0024]** The variable delay element is varied according to a certain schedule. Enough time has to be provided between successive changes of the delay, such that the generation speed can be reliably determined. Here is described one possible delay schedule algorithm that is considered the best mode, but there are many other possible algorithms that could be used.

**[0025]** The delay values and the corresponding speed during the last K (a hundred or so) different delay settings are stored and updated with the measured speed from each new delay setting.

**[0026]** The delay changes in the schedule are relative from the stored optimum delay value. A pseudo-random sequence of Gaussian-like distribution is used. Small delay changes in each direction are often used, with larger increments being less and less likely to be applied. This keeps the delay around its ever-changing optimum value with occasional experiments of further away values. If these changes are successful, the center of subsequent changes moves there. If a delay change is unsuccessful (resulting in slower generation of random bits) it will be forgotten, with the next adjustment of the delay being from the previous optimum.

**[0027]** The scheduling algorithm needs to store the last K measured generation-speed values in a queue (ring-buffer, for example) in such a manner that, the current maximum is always available. A heap or priority queue data structure can perform quite efficiently for this purpose.

**[0028]** The following algorithm in MATLAB code is provided as one way that a delay scheduling algorithm may look. A computer readable medium may contain this code, or it could be loaded, for example, into a microprocessor. However, an artisan understands that there can be other algorithms, either written in MATLAB or other languages within the spirit of the invention and the scope of the appended claims that perform these functions. An artisan also understand that the following has REMARKS following the "%" symbol to help the reader understand what is happening in the algorithm.

```
qlen = 100;    % Queue length
dmax = 256;    % #delay values
```

```
dsig = sqrt(dmax);      % standard deviation of steps
speed = zeros(1,qlen); % start with speed 0
delay = zeros(1,qlen) + dmax/2;        %start with median delay
i = 1;    % insertion point in the queue
while 1% infinite loop to keep max speed
dstep = randn*dsig;    % steps of normal distribution
dstep = sign(dstep)*ceil(abs(dstep));% ensure |step| > 0
[smx,imx] = max(speed); %last max and its index in queue
dmx = delay(imx);
dly = max(1,min(256,dmx+dstep));  % next try
spd = GetSpeed(dly);  % set delay, get speed
delay(i) = dly; % store trial results
speed(i) = spd;
i = i + 1;          % move insertion point in queue
if i > qlen, i = 1; end
end.
```

**[0029]** One interpretation of the above code as a series of sub-steps is as follows:

(i) setting a queue length at a predetermined value;

(ii) (ii) setting a predetermined number of delay values;

(iii) designating a standard deviation (dmax) of steps;

(iv) starting with speed of 0;

(v) starting with a median delay;

(vi) setting an insertion point in the queue while keeping an infinite loop at maximum speed;

(vii) designating a number of steps of normal distribution;

(viii) ensuring that |step| > 0;

(ix) obtaining a last maximum speed and its index in the queue;

(x) setting the delay as imax;

(xi) repeating for next next delay value (from 1 to 256);

(xii) setting delay (dly) and getting speed (spd);

(xiii) storing trial results of speed and updating a variable delay unit used for random number generation;

(xiv) moving/increasing insertion point i in the queue by 1;

(xv) if the insertion point i > que length, and i = 1, then ending the routine;

(xvi) go to step (xi).

**[0030]** Fig. 4 is a general overview of a method according to the present invention.

**[0031]** At step 400, the frequency of the random bit generator is measured. As previously discussed, this frequency can be measured via a microprocessor, or with hardware such as a counter multiplied with a weight at every clock cycle and incremented each time a random bit is generated.

**[0032]** At step 405 it is determined whether a predetermined amount of time has passed since the last time the variable delay unit (310 shown in Fig. 3) has been tuned. The reason for the predetermined amount of time is to ensure that the generation speed can be reliably determined. A person of ordinary skill in the art also understands that this step could be interchanged with the first step, and no measuring of the frequency could occur until the predetermined threshold of time has passed.

**[0033]** If it is determined that the predetermined threshold of time has not passed, the method reverts to the first step 405. However, if it has been determined that the threshold has passed, at step 410 it is determined whether frequency is at a maximum. If the maximum speed is being maintained, there is no reason to adjust the variable delay element, and step 410 is repeated until it is determined that the measured speed is not the maximum speed.

**[0034]** At step 415, the variable delay is dynamically tuned by a predetermined value to optimize the speed, and the time is reset with regard to how long it has been since the last update of the variable delay unit. The predetermined amount of speed value can be stored in a ring-buffer, queue, table, dynamically updated table, etc. Ideally, it is an updated storage wherein a value can be picked that optimizes the possibility that maximum speed will be reached. The method then reverts to step 400 and the new speed is measured. A queue or

table could be updated after the predetermined threshold has been reached regarding the degree of variable delay and the speed of the random bit generation.

**[0035]** There are various modifications to the foregoing invention that could be performed by a person of ordinary skill in the art that would lie within the spirit of the invention and the scope of the appended claims. For example, the type of algorithms used, the language in which the algorithms are written, equivalent logic gates other than NAND gates, variations in code or types of code used, defined weight, etc., etc., are all within the teachings of the instant invention. The arrangement of the variable delay and fixed delay units, and the measurement module could be arranged differently than shown in Fig. 3.